数理解析研究所講究録
556巻 1985年 231-239

A Remark on Solving the Set-Partitioning Problem by Dual All Integer Algorithm

Kakuzo Iwamura
Mathematics
Josai University
Sakado, Saitama

Abstract

A careful consideration when one solves the set-partitioning problem by dual all integer algorithm is presented.
It saves both computing time and memory size.

[1]. Introduction

A Set-Partitioning Problem,

minimize $x_0=\sum_{j=1}^{n}c_jx_j$

subject to $\sum_{j=1}^{n}a_{ij}x_j=1\,(1\leq i\leq m)$, x_j:binary$(1\leq j\leq n)$, (1.1)

where c_j positive integer, $a_{ij}=0$ or 1 can be solved by Dual All Integer Algorithm[1,2]. Salkin & Koncal[4,5,6] transformed this problem to the Set-Covering Problem,

maximize $u_0=\sum_{j=1}^{n}(c_j+Lh_j)(-x_j)$

subject to $\sum_{j=1}^{n}a_{ij}x_j\geq 1\,(1\leq i\leq m)$, x_j:binary$(1\leq j\leq n)$, (1.2)

where integer L is greater than $\sum_{j=1}^{n}c_j$, $h_j=\sum_{i=1}^{m}a_{ij}$ [1,3] and solved the original Set-Partitioning Problem successfully.

Setting $x_{n+i}=\sum_{j=1}^{n}a_{ij}x_j-1\,(1\leq i\leq m)$, they applied Dual All Integer Algorithm to the dual feasible all integer tableau as follows[1,2];

$$\begin{array}{lccccc}
 & 1 & -x_1 & -x_2 & \cdots & -x_n \\
u_0 & 0 & c_1+Lh_1 & c_2+Lh_2 & \cdots & c_n+Lh_n \\
x_{n+1} & -1 & -a_{11} & & & -a_{1n} \\
x_{n+2} & -1 & & & & -a_{2n} \\
\vdots \; = & & & & & \\
x_{n+m} & -1 & -a_{m1} & & & -a_{mn}
\end{array} \qquad (1.3)$$

Maximum tableau size could grow as large as $(m+n+2)(n+1)$ including a cut row.

[2]. Another Transformation

Let's consider another transformation which transforms (1.1) to

maximize $v_0 = -\sum_{j=1}^{n} c_j x_j$

subject to $\sum_{j=1}^{n} a_{ij} x_j = 1 \ (1 \leqq i \leqq m), x_j \geq 0$ integer $(1 \leqq j \leqq n)$, (2.1)

where $v_0 = -x_0$.

Let M be any integer greater than the minimal value x_0 of (1.1), for example $M = \sum_{j=1}^{n} c_j + 1$, then we see that

$$v(2.1) > -M \tag{2.2}$$

as $v(1.1) = -v(2.1)$, where $v(P)$ denotes the optimal value of the 0-1 integer programming problem (P).

Consider one more problem such as

maximize $w_0 = -\sum_{j=1}^{n} c_j x_j - M \sum_{i=1}^{m} x_{n+i}$

subject to $\sum_{j=1}^{n} a_{ij} x_j - x_{n+i} = 1 \ (1 \leqq i \leqq m)$, $x_u \geqq 0$ integer $(1 \leqq i \leqq m+n)$. (2.3)

We easily see that the following properties hold.

<u>Property a</u>; (2.3) has a dual feasible integer solution $x_j = 0 \ (1 \leqq j \leqq n)$, $x_{n+i} = -1$ $(1 \leqq i \leqq m)$ with the same dual feasible all integer tableau as (1.3), u_0, L replaced by w_0, M.

<u>Property b</u>; (2.1) has a feasible integer solution if and only if (2.3) has a feasible integer solution whose objective function value w_0 is greater than $-M$.

Property c; $v(2.3) \geqq -\sum_{j=1}^{n} c_j - mM$

From these properties, we can obtain an optimal integer solution of (2.3) after finite iterations of Dual All Integer Algorithm. Moreover we have,

$$v(2.3)\begin{cases} > -M & \text{iff every optimal integer solution of (2.3) is an optimal} \\ & \quad \text{integer solution of (2.1) \& } v(2.3) = v(2.1), \\ \leqq -M & \text{iff (2.3) is infeasible,} \end{cases}$$

so that we get the next Procedure d.

Procedure d; Every time any variable x_u $(n+1 \leqq u \leqq n+m)$ becomes nonbasic in the course of dual pivoting, we can drop x_u and its corresponding column from the tableau.

[3]. Example

I quote Dual All Integer Algorithm from [1].

Step 0; (Preparation) Prepare simplex tableau,

$$x_{B_i} = y_{i0} + \sum_{j\in R} y_{ij}(-x_j), \quad (0 \leqq i \leqq m), \tag{3.1}$$

where $x_{B_0} = x_0$ = objective function value, x_{B_i} $(1 \leqq i \leqq m)$ are basic variables, x_j $(j\in R)$ are nonbasic variables. A vector $v \neq 0$ is called lexicographically positive if its first nonzero component is positive. We use rotation $v \overset{L}{>} 0$ to denote v lexicographically positive. We use y_j to denote the j-th column of the simplex tableau (3.1). Simplex tableau (3.1) is called dual feasible if $y_j \overset{L}{>} 0$ for all $j\in R$, all integer if y_{ij} $(0 \leqq i \leqq m,\ 0 \leqq j \leqq n)$ are all integers. $[u]$ denotes the largest integer less than or equal to u.

Step 1: (Initialization) Begin with a dual feasible all integer tableau (3.1). Go to Step 2.

Step 2: (Test for optimality) If the solution is primal feasible, it is optimal to (3.1). STOP. If not, go to Step 3.

Step 3: (Cutting and pivoting) Choose a source row $(i \neq 0)$ in the tableau with $y_{i0} < 0$, say $i=r$. The topmost row with $y_{i0} < 0$ must be chosen at least periodically. Select the lexicographically smallest column with $y_{rj} < 0$, say $j=k$, as the pivot column. Compute $\bar{h}$ by

$$\bar{h} = \min_{j\in R_r} \frac{\bar{M}_j}{y_{rj}}$$

where $R_r = \left\{ j\in R \mid y_{rj} < 0 \right\}$, $\bar{M}_k = -1$, $\bar{M}_j = \min\left\{ u \mid y_j + u y_k \overset{L}{>} 0,\ u \text{ integer} \right\}$ for $j\in R_r \setminus \{k\}$.

If $\bar{h}=1$, execute one dual simplex iteration with pivot element y_{rk}.

If $\bar{h}<1$, adjoin the cut

$$s = [\, hy_{r0}] + \sum_{j\in R} [\, hy_{rj}](-x_j)$$

with $h = \bar{h}$, to the bottom of the tableau. Execute a dual simplex iteration with s as the departing variable and x_k as the entering variable. In any case, if x_k is a slack from a cut, delete the x_k row. Return to Step 2.

To see the power of Procedure d, we take the Example from [1, page 315].

minimize $3x_1+7x_2+5x_3+8x_4+10x_5+4x_6+6x_7+9x_8$

$$\begin{aligned}
x_1+x_2 &= 1\\
x_3+x_4+x_5 &= 1\\
x_5+x_6+x_7 &= 1\\
x_7+x_8 &= 1\\
x_2+x_4+x_6 &= 1
\end{aligned}$$

We start with dual feasible all integer tableau (3.2) which is obtained through replacing u_0, L by w_0, $M = \sum_{j=1}^{8} c_j +1 = 53$.

$$\begin{array}{lrrrrrrrrr}
 & 1 & \overset{\overset{k}{\downarrow}}{-x_1} & -x_2 & -x_3 & -x_4 & -x_5 & -x_6 & -x_7 & -x_8 \\
w_0 & 265 & 56 & 113 & 58 & 114 & 116 & 110 & 112 & 62 \\
r)\,x_9 & -1 & \enclose{circle}{-1} & -1 & & & & & & \\
x_{10} & -1 & & & -1 & -1 & -1 & & & \\
x_{11} = & -1 & & & & & -1 & -1 & -1 & \\
x_{12} & -1 & & & & & & & -1 & -1 \\
x_{13} & -1 & & -1 & & -1 & & -1 & &
\end{array} \qquad (3.2)$$

$r=1$, $R_r = \{1,2\}$, $k=1$, $\bar{M}_1= -1$, $\bar{M}_2= -2$, $y_{rk}= -1$(circled) gives $\bar{h}=1$. Pivoting on y_{rk} makes x_1 basic, x_9 nonbasic so that we may drop x_9 column from the new tableau (3.3).

$$\begin{array}{lrrrrrrrr}
 & 1 & -x_2 & \overset{\overset{k}{\downarrow}}{-x_3} & -x_4 & -x_5 & -x_6 & -x_7 & -x_8 \\
w_0 & 209 & 57 & 58 & 114 & 116 & 110 & 112 & 62 \\
x_1 & 1 & 1 & & & & & & \\
r)\,x_{10} & -1 & & \enclose{circle}{-1} & -1 & -1 & & & \\
x_{11} = & -1 & & & & -1 & -1 & -1 & \\
x_{12} & -1 & & & & & & -1 & -1 \\
x_{13} & -1 & -1 & & -1 & & -1 & &
\end{array} \qquad (3.3)$$

$r=2$, $k=2$, $\bar{M}_2= -1$, $\bar{M}_3= -1$, $\bar{M}_4= -2$, $y_{rk}= -1$(circled) gives $\bar{h}=1$. Pivoting on y_{rk} makes x_3 basic, x_{10} nonbasic so that we may drop x_{10} column from the next tableau (3.4).

$$\begin{array}{l|rrrrrrr}
 & 1 & -x_2 & -x_4 & \overset{k}{-x_5} & -x_6 & -x_7 & -x_8 \\
w_0 & 151 & 57 & 56 & 58 & 110 & 112 & 62 \\
x_1 & 1 & 1 & & & & & \\
x_3 & 1 & & 1 & 1 & & & \\
r)\, x_{11} = & -1 & & & \enclose{circle}{-1} & -1 & -1 & \\
x_{12} & -1 & & & & & -1 & -1 \\
x_{13} & -1 & -1 & -1 & & -1 & &
\end{array} \qquad (3.4)$$

Doing in this way, i.e.,

x_5 basic, x_{11} nonbasic drop x_{11} column;
x_7 basic, x_{12} nonbasic drop x_{12} column;
x_6 basic, x_{13} nonbasic drop x_{13} column;
x_4 basic, x_5 nonbasic drop none,

we get final tableau (3.5) which is optimal.

$$\begin{array}{l|rrrr}
 & 1 & -x_2 & -x_5 & -x_8 \\
w_0 & -17 & 1 & 4 & 4 \\
x_1 & 1 & 1 & & \\
x_3 & 0 & -1 & 2 & -1 \\
x_4 = & 1 & 1 & -1 & 1 \\
x_7 & 1 & & & 1 \\
x_6 & 0 & & 1 & -1
\end{array} \qquad (3.5)$$

As $v(3.5)=-17 > -53$, we see that $x_1=x_4=x_7=1$, $x_j=0$ (otherwise), $x_0=17$ is an optimal solution. Final tableau size is half as large as the original. We also do away with needless calculations for the deleted columns.

References

[1] Garfinkel, R.S., and Nemhauser, G.L., "Integer Programming", John Wiley & Sons, 1972

[2] Hu, T.C., "Integer Programming and Network Flows", Addison-Wesley, 1970

[3] Lemke, C., Salkin, H., and Spielberg, K., "Set Covering by Single Branch Enumeration with Linear Programming Subproblems", Oper. Res. 19 (1971), 998-1022

[4] Salkin, H.M., and Koncal, R., "A Pseudo Dual All-integer Algorithm for the Set Covering Problem", Technical Memorandum No.204, Nov. 1970, Case Western Reserve University

[5] Salkin, H.M., and Koncal, R., "A Dual All-integer Algorithm (in Revised Simplex Form) for the Set Covering Problem", Technical Memorandum No.250, Aug. 1971, Case Western Reserve University

[6] Salkin, H.M., and Koncal, R., "Set Covering by an All Integer Algorithm: Computational Experience", J.ACM 20 (1973), 189-193